ROSdev: GUI-Aided Software Development in ROS

Filip Müllers, Dirk Holz, Sven Behnke
University of Bonn

SDIR VII at ICRA 2012
St. Paul, Minnesota, USA—May 14th, 2012

universitätbonn ais

Introduction

ROSdev: GUI-Aided Software Development in ROS:

- What is **ROS**,
- what is **missing in ROS**?, and
- how are we (trying to) **fill this gap**?

ROSdev and xROSdev (formerly rxDeveloper)
`http://code.google.com/p/rxdeveloper-ros-pkg`

universitätbonn ais

Introduction: *What is ROS?*

"ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license."—**ros.org**

ROS provides

- an infrastructure for **communication** between components in a robot control architecture or *"nodes"*, and
- a set of **tools**, e.g., for monitoring **...**

- a **community** developing and publishing (open source) software packages.

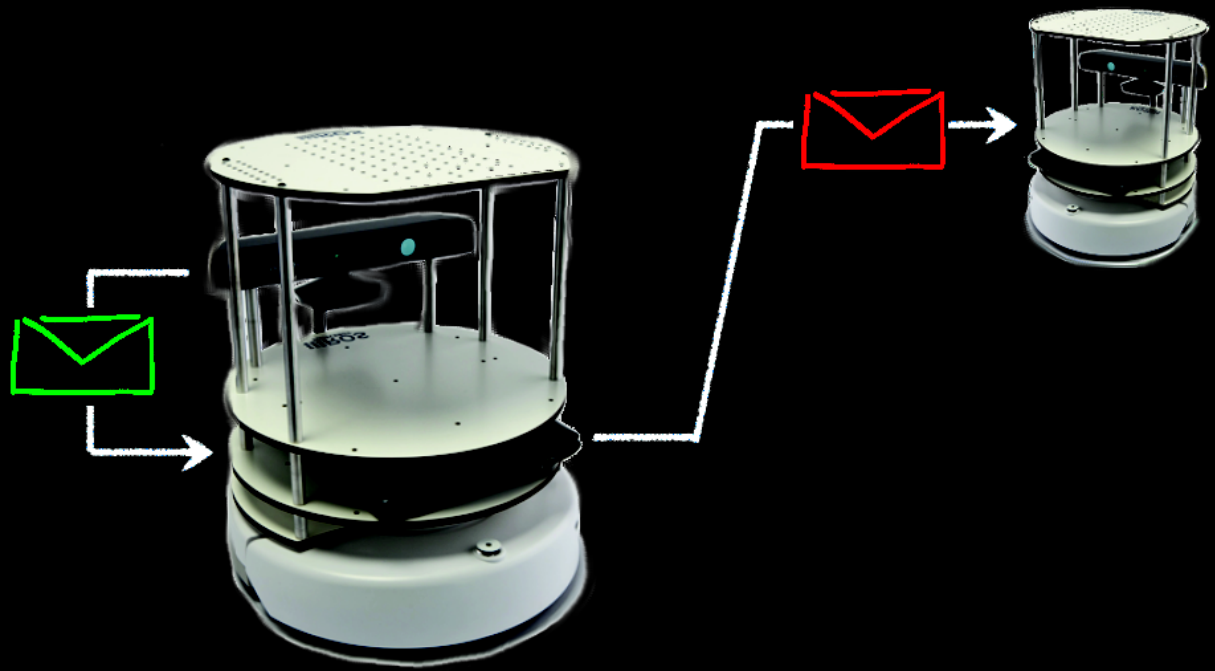
universitätbonn ais

Introduction: *What is ROS?*

Communication (*"messages"*)

- between different robots and
- between individual components (*"nodes"*).

→ in principle just **"messages between nodes"**



Publisher-subscriber-architecture

\+ *"services"* for RPC-like request/response message pairs.

universitätbonn ais

Introduction: *What is ROS?*

Communication

- publishing data to **advertised topics**
- retrieving data from **subscribed topics**
- communication by connecting topics (**remapping**)

Famous example (using the Unix-like command line tools)

```
$ rostopic pub -r 1 /info std_msgs/String 'ROSdev is great!'


$ rostopic echo /info
data: ROSdev is great!
---
data: ROSdev is great!
---
```

universitätbonn ais

Introduction: *What is ROS?*

Communication: Messages

- simple message specification files
- code (e.g. C++ and Python) is automatically generated.

Example: Pose (position and orientation in 3D)

- Pose.msg

```
# A representation of pose in free space, composed of
position and orientation.
Point position
Quaternion orientation
```

- Point.msg

```
# This contains the position of a point in free space
float64 x
float64 y
float64 z
```

universitätbonn ais

Introduction: *What is ROS?*

Tools

- (Almost) everything can be done using a set of simple command line tools.
- A few other tools have graphical interfaces and allow, e.g., for visualization.

Examples (and what is shipped with ROS)

- Tools for managing packages (and stacks)

  `rospack/rosstack, roscreate-pkg/roscreate-stack, rosdep`
- Navigating to packages, stacks, and files

  `rosmake, roscd, rosls, rosed, ...`
- Retrieving information about nodes and messages

  `rosnode, rosmsg`

universitätbonn ais

Introduction: *What is ROS?*

Examples (and what is shipped with ROS)

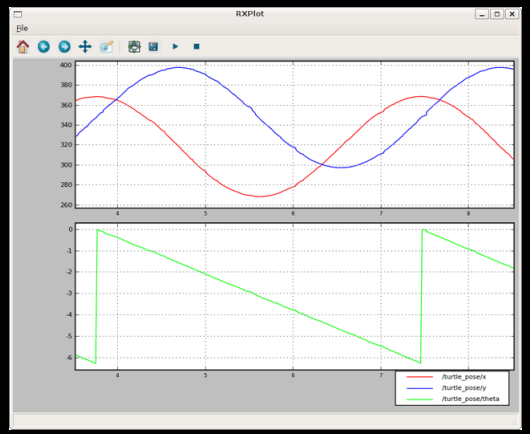
- Retrieving and setting parameters

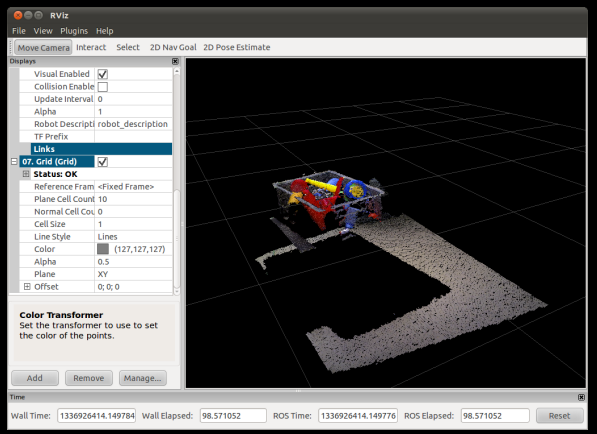
  `rosparam`
- Listing and calling services

  `rosservice`
- Listing and retrieving information from/about topics

  `rostopic`
- Visualizing data

  `rxplot, rviz`





universitätbonn ais

Introduction: *What is ROS?*

Tools: Running nodes and whole computation graphs

- Running a single component / node

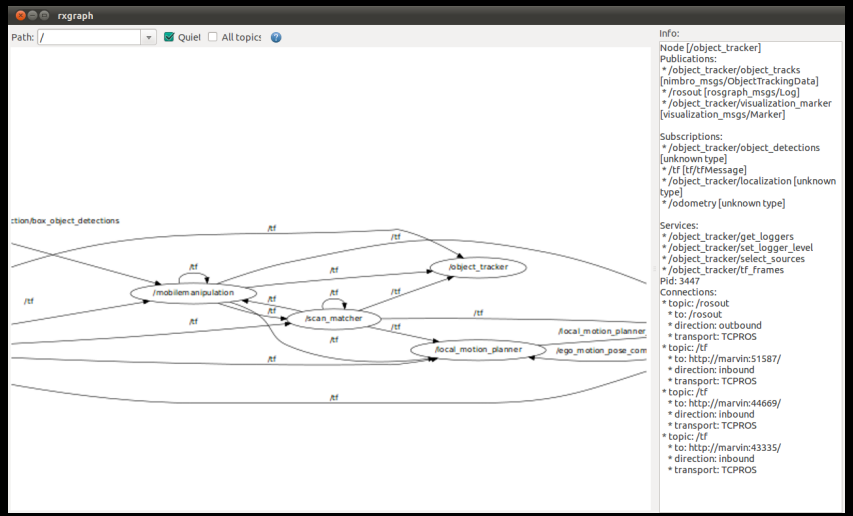
  `rosrun`

- Running a graph of connected components (specified in *"launch"* files)

  `roslaunch`

- Visualizing the graph of components

  `rxgraph (and rosgraph)`



universitätbonn ais

Introduction: *What is ROS?*

launch **files** represent graphs of

- nodes,
- connection between nodes, and
- parameters.

```
1   <node name="hokuyo" pkg="hokuyo" type="hokuyo">
2     <remap from="scan" to="base_laser_scan" />
3     <param name="frameid" value="base_laser" />
4     <param name="calibrate_time" value="false" />
5   </node>
6   <node name="laser_slam" pkg="slam" type="slam">
7     <remap from="~scan" to="base_laser_scan"
8   </node>
9   <include file="teleop.launch">
```

universitätbonn ais

Introduction: *What is missing ROS?*

You can:

- specify message types (in .msg files),
  and generate code from these specifications.
- specify graphs of nodes (in .launch files),
  and run/launch them.
- visualize graphs (not interactively).

You cannot:

- specify nodes (in .node files),
  and generate code from these specifications.
  → **ROSdev's node specification files**
- visually edit computation graphs.
  → **xROSdev**

universitätbonn ais

ROSdev: Node *Specification Files*

ROSdev's Node Specification Files

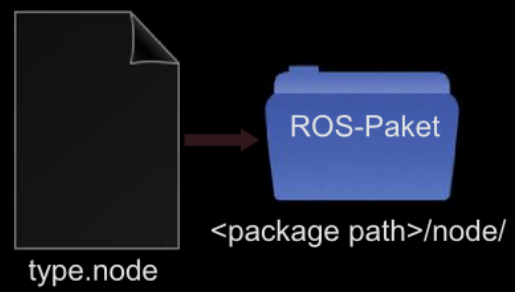
- **Purpose:**
  Description of components (*nodes* and *nodelets*)
  - Package and type information
  - Publications
  - Subscriptions
  - Services
  - Parameters (with ranges, defaults, etc.)
- **Format:**
  YAML—intuitive syntax and semantics
- **Uses:**
  - generating code stubs and
  - container of interface information as is

Right now there is no such structure in ROS,
interfaces are defined manually when writing the code.

universitätbonn ais

ROSdev: Node *Specification Files*

```
1   type: binary_node
2   package: node_package
3   subscriptions:
4    - topic: first/topic_name
5      type: topic_type
6    - topic: second/topic_name
7      type: topic_type
8    - ...
9   publications:
10   - topic: first/topic_name
11     type: topic_type
12   - topic: second/topic_name
13     type: topic_type
14
15  services:
16   - name: first/service
17     type: service_type
18   - name: second/service
19     type: service_type
20   - ...
21  parameters:
22   - name: firstParameter
23     type: string
24     default: "defaultText"
25   - name: ~anotherParameter
26     type: double
27     default: 0.0
28     range: [-1.0,1.0]
29   - ...
```



universitätbonn ais

ROSdev: Node *Specification Files*

Why are these specification files needed?

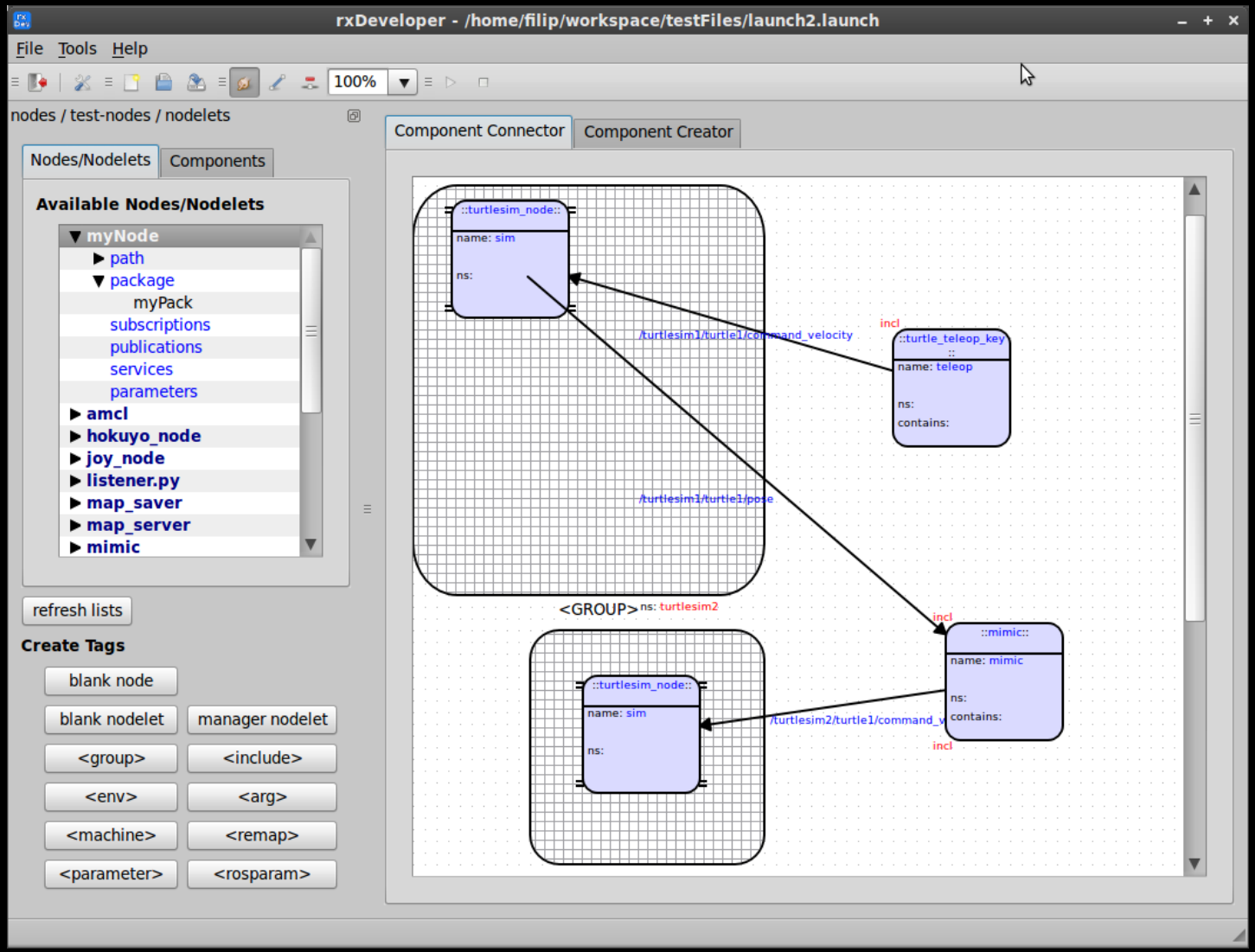
- ROS tools cannot retrieve all the information about a node.
- Specification files simply contain all the information.

- Specification files are not necessary for xROSdev—our graphical tool for aiding software development in ROS, but very useful and support the user in many ways:
  - New node tags can be created by dragging and dropping
  - When connecting nodes, lists of all possible subscriptions/publications pop up
  - For parameters, a list of all possible parameters pops up
  - parameter ranges and automated checking of values
  - . . .

universitätbonn ais

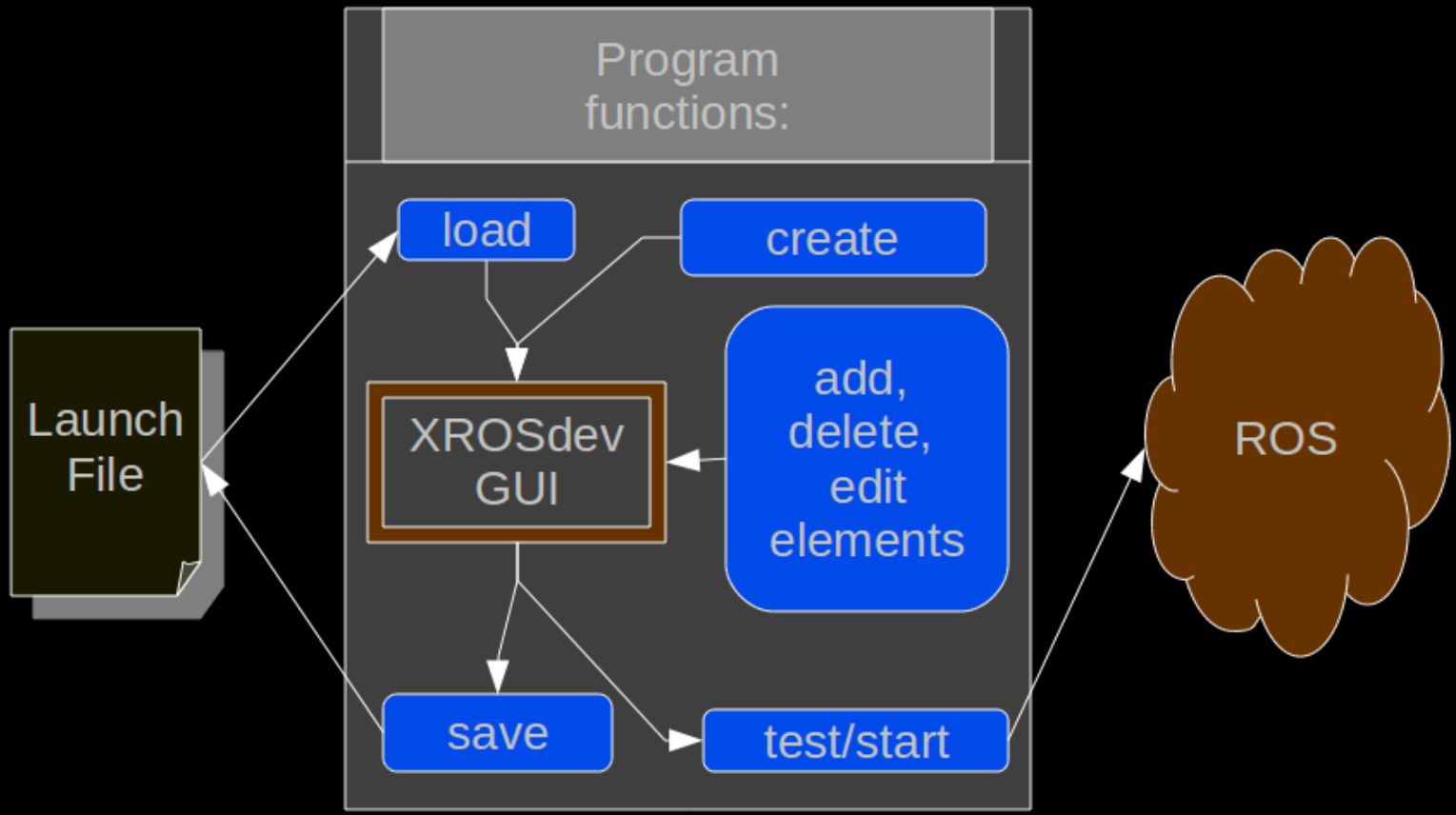
xROSdev: Graphs and *Launch Files*

xROSdev = *"GUI for editing launch files"*



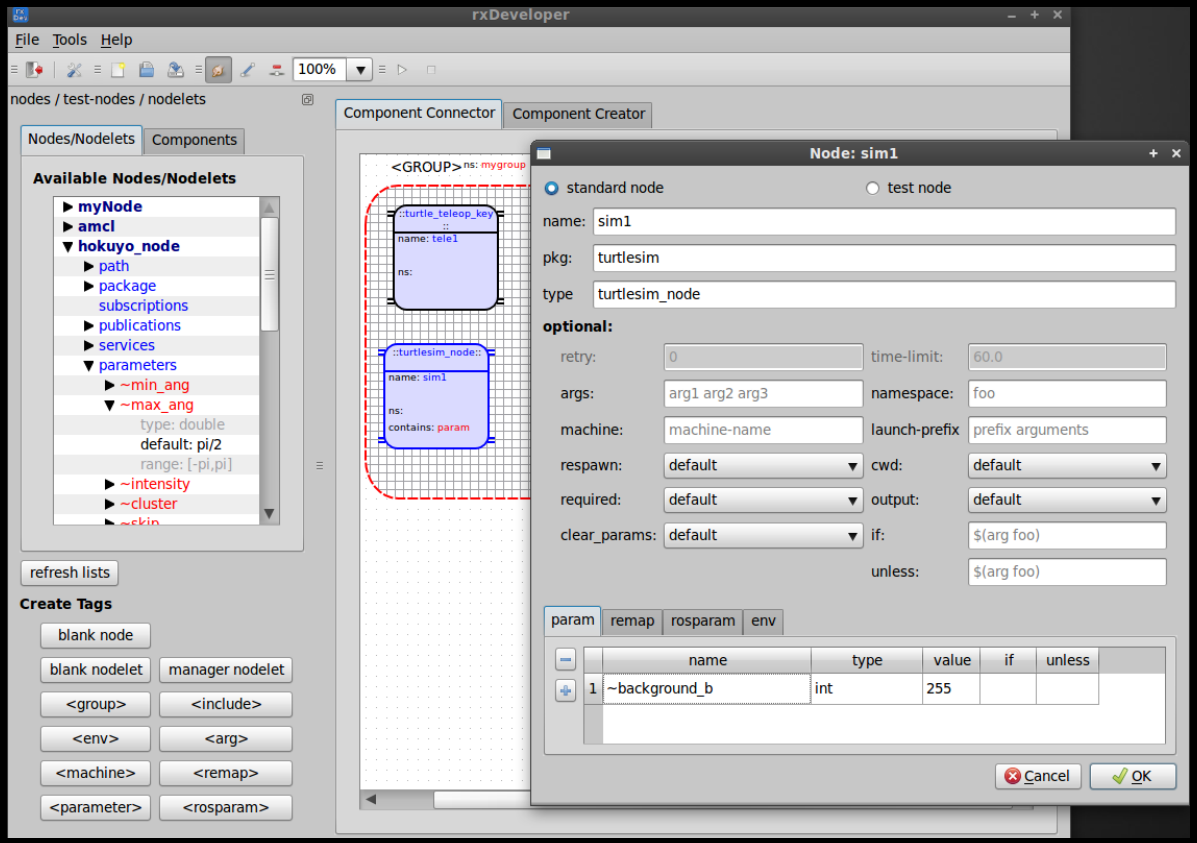
xROSdev: Graphs and *Launch Files*

Overview



universitätbonn ais

xROSdev: Graphs and *Launch Files*

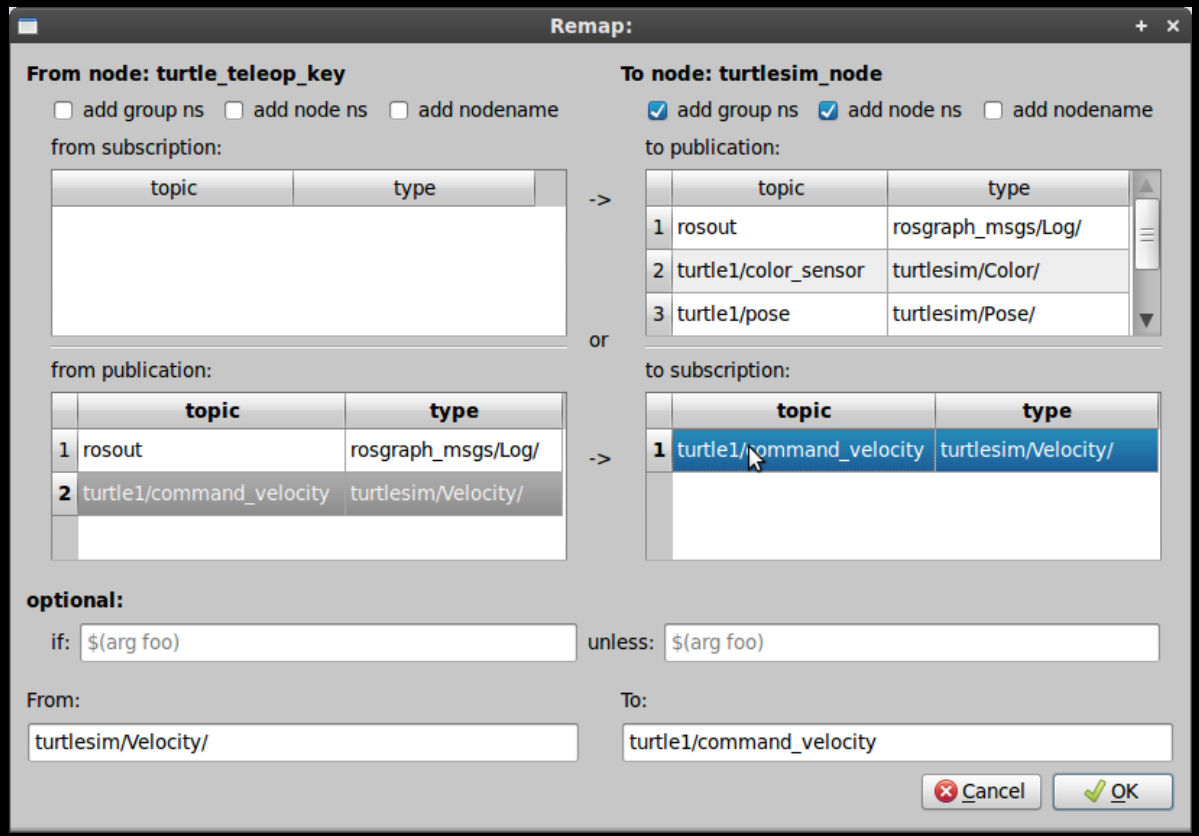


In the **component connector**, you can easily

- drag&drop, insert&remove nodes, and
- connect nodes by means of topic remaps.

universitätbonn ais

xROSdev: Graphs and *Launch Files*



In the **remap editor**, you can easily

- select among available topics (with specification files)
- manually fill in topic names (without specification files)

universitätbonn ais

xROSdev: Graphs and *Launch Files*

Extending the former **launch file** format.

- For being able to recover the setup of nodes in the GUI
- 100% compatible by using xml comments to encode positions and dimensions on screen.

```
1  <launch>
2    <!--Created with rxDev-->
3    <group ns="group1">
4      <!--x="80" y="70" width="200" height="200"-->
5      <node name="sim" pkg="turtlesim" type="turtlesim_node">
6        <!--x="52" y="53"-->
7      </node>
8    </group>
```

universitätbonn ais

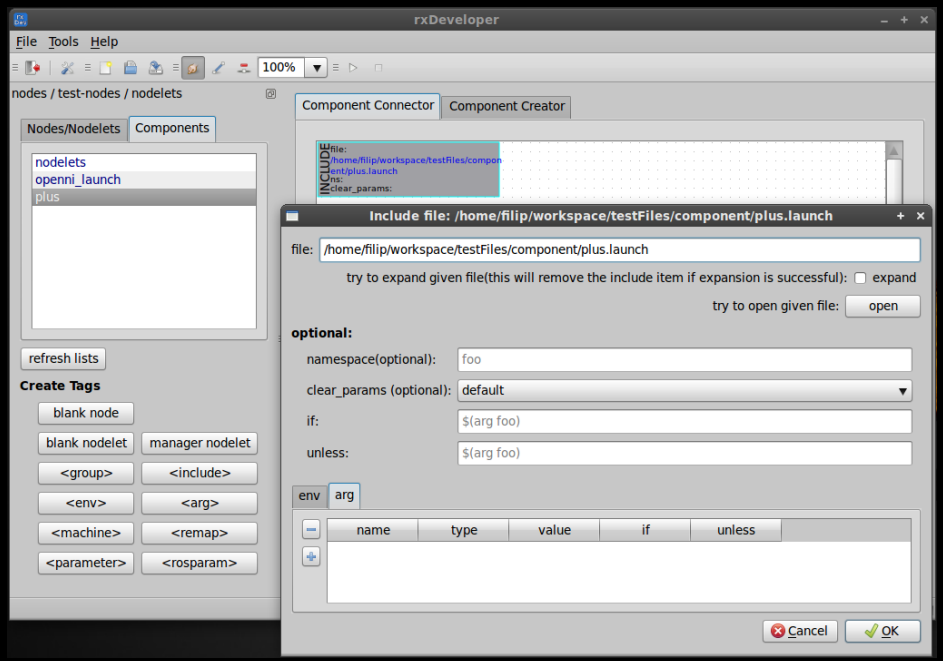
xROSdev: Graphs and *Launch Files*

Including other launch files

- Commonly used node combinations can be group and stored in individual launch files.

In xROSdev you can:

- add other launch files as a block (itself containing another launch file)
- completely copying the complete content into the currently edited graph.



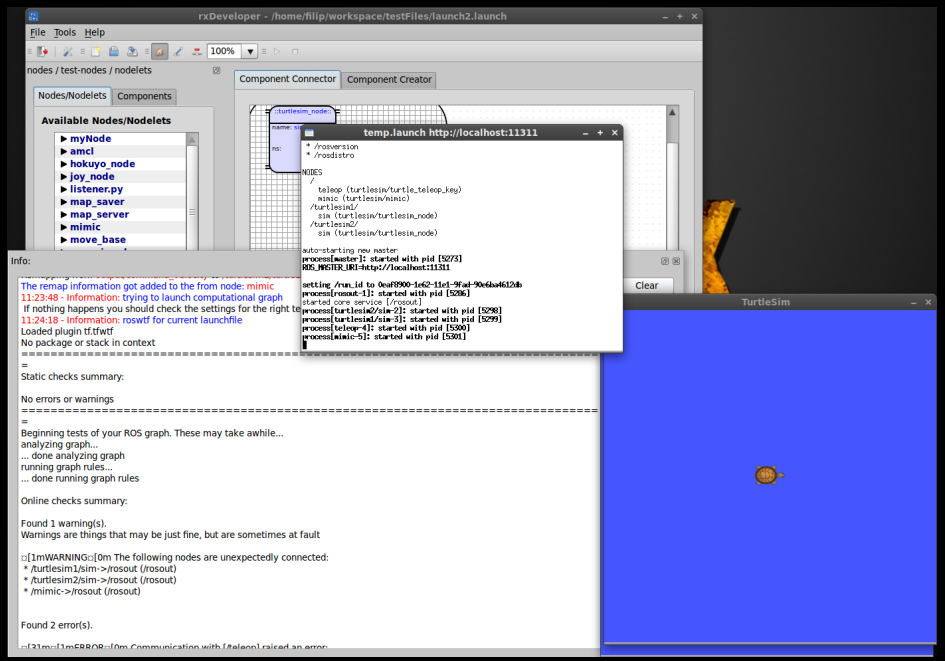
universitätbonn ais

xROSdev: *Testing and Debugging*

Starting, stopping and monitoring execution

- The main toolbar feature play and stop buttons to directly start the currently edited launch file.
- You can monitor the graph's execution using the usual tools (again directly started out of xROSdev)

- rxgraph
- rviz
- rxloggerlevel
- rxconsole
- roswtf
- . . .

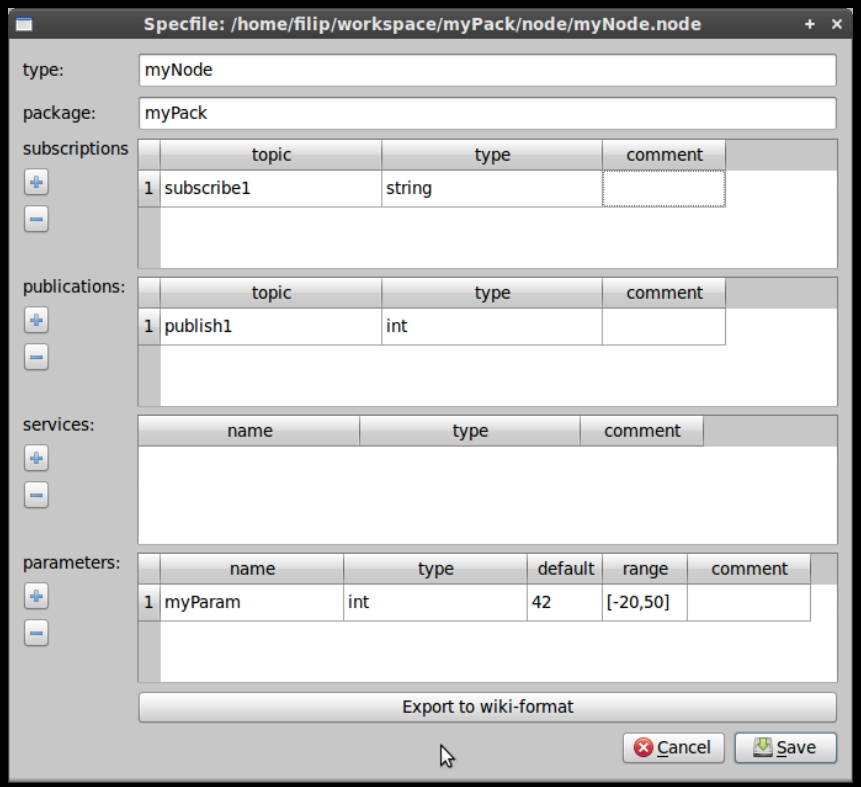


universitätbonn ais

xROSdev: *Creating new nodes*

A **specification file editor** allows for

- easily creating new nodes and
- editing existing ones.
- Generated code stubs are updated and overwritten.

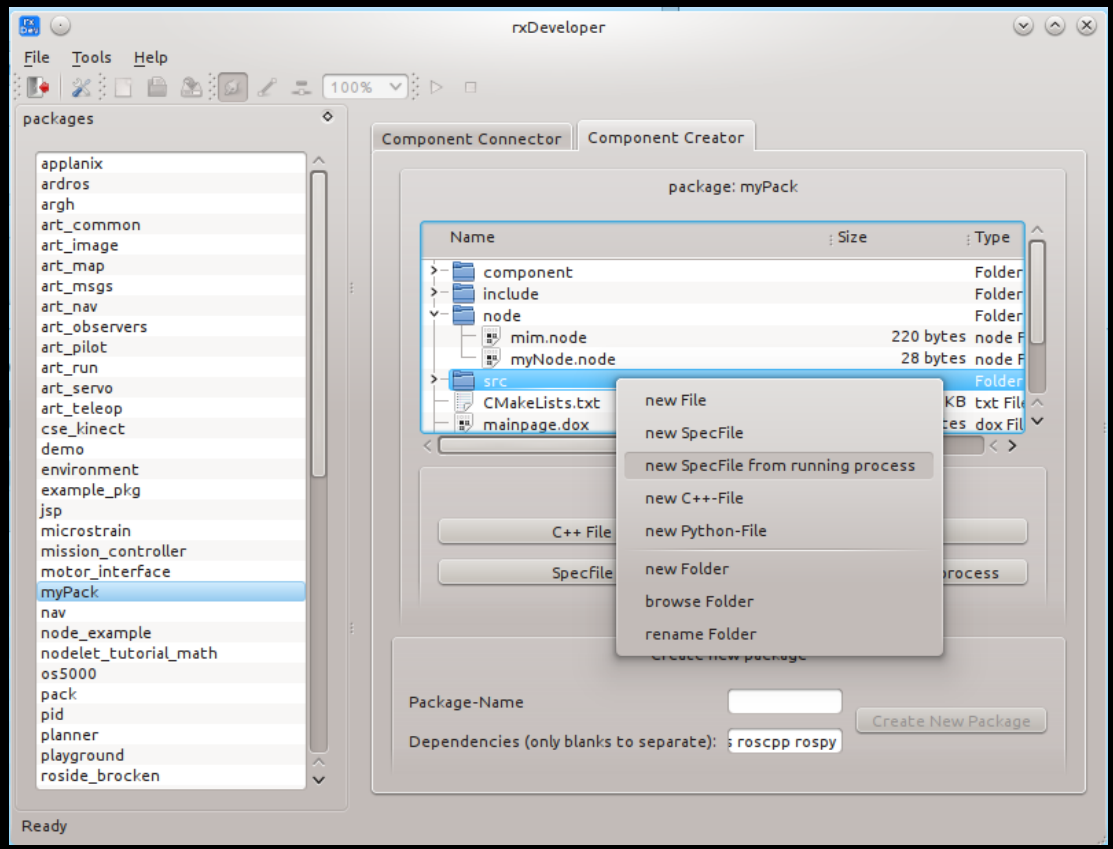


universitätbonn ais

xROSdev: *Creating new packages*

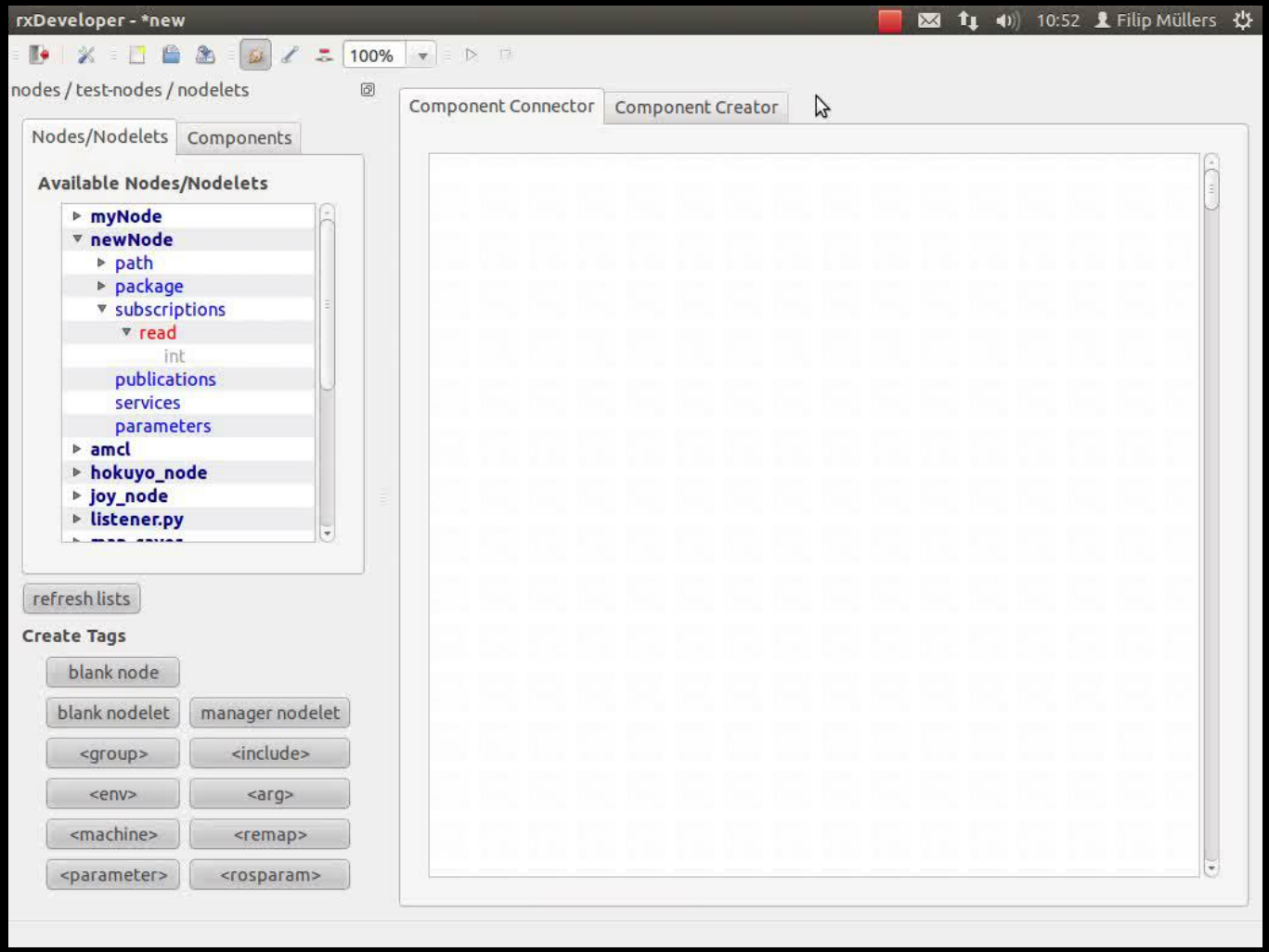
The **component creator** allows

- browsing existing packages
- creating new packages
- creating files (message/service/node specifications, . . .)



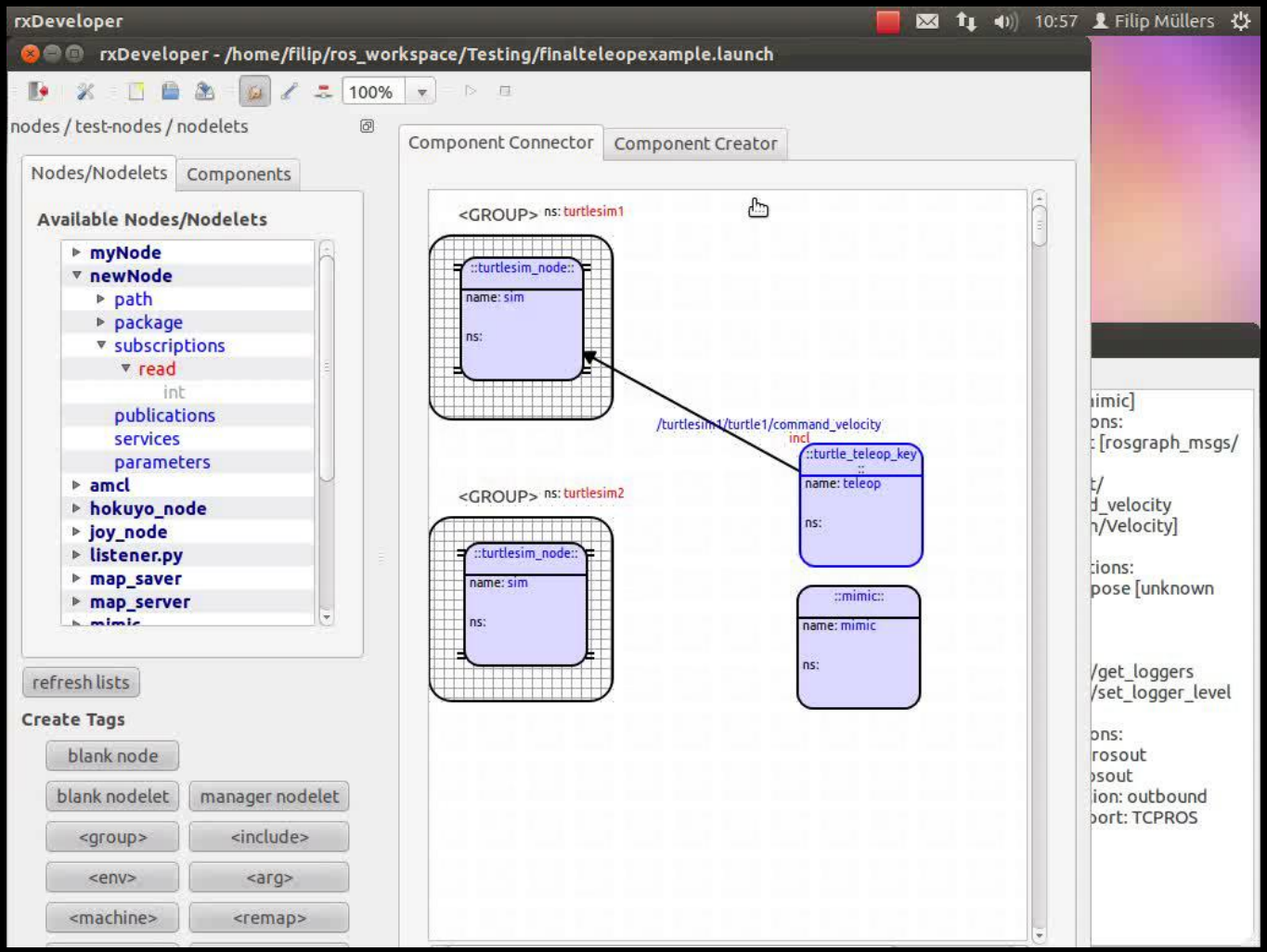
xROSdev: *Node specification → code*

Generate code stubs form a node specification file.



xROSdev: *Code → node specification*

Import node information and specification from a running node.



universitätbonn ais

ROSdev/xROSdev: *"Conclusions"*

We have presented *"yet another ROS tool"* for

- aiding the software development process in ROS,
- allowing graphical editing of launch files (as opposed to manually editing xml),
- allowing the generation of code for ROS nodes (and not only for messages and services)

universitätbonn ais

ROSdev/xROSdev: *"Outlook"*

Future and Ongoing Work

- ROSdev is currently under development,
- feel free to **join the team**.
- Currently addressed topics:
  - **Full integration into ROS** (node specification files)
  - Shipping ROS with xROSdev for GUI-aided development
  - Variety of templates for generating code stubs

Thank you for your attention!

Any questions?

visit

`http://code.google.com/p/rxdeveloper-ros-pkg`